Date=24/11/2020
Lecture By=Manish Mahant
Subject ⇒React Lifecycles

IN PREVIOUS LECTURE **(QUICK RECAP) Date-23/11/2020**	**In Today's Lecture (Overview)**
Lists and Keys Rendering Multiple Components Basic List Component Keys Extracting Components with Keys Keys Must Only Be Unique Among Siblings Embedding map() in JSX	React Lifecycle Lifecycle of Components Mounting Unmounting

React Lifecycle

Lifecycle of Components

Each component in React has a lifecycle which you can monitor and manipulate during its three main phases.

The three phases are: Mounting, Updating, and Unmounting.

Mounting

Mounting means putting elements into the DOM.

React has four built-in methods that gets called, in this order, when mounting a component:

1. `constructor()`
2. `getDerivedStateFromProps()`
3. `render()`
4. `componentDidMount()`

The `render()` method is required and will always be called, the others are optional and will be called if you define them.

constructor

The `constructor()` method is called before anything else, when the component is initiated, and it is the natural place to set up the initial `state` and other initial values.

The `constructor()` method is called with the `props`, as arguments, and you should always start by calling the `super(props)` before anything else, this will initiate the parent's constructor method and allows the component to inherit methods from its parent (`React.Component`).

Example:

The `constructor` method is called, by React, every time you make a component:

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  render() {
    return (
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
    );
  }
}
ReactDOM.render(<Header />, document.getElementById('root'));
```

getDerivedStateFromProps

The `getDerivedStateFromProps()` method is called right before rendering the element(s) in the DOM.

This is the natural place to set the `state` object based on the initial `props`.

It takes `state` as an argument, and returns an object with changes to the `state`.

The example below starts with the favorite color being "red", but the `getDerivedStateFromProps()` method updates the favorite color based on the `favcol` attribute:

Example:

The `getDerivedStateFromProps` method is called right before the render method:

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  static getDerivedStateFromProps(props, state) {
    return {favoritecolor: props.favcol };
  }
  render() {
    return (
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
    );
  }
}

ReactDOM.render(<Header favcol="yellow"/>,
document.getElementById('root'));
```

render

The `render()` method is required, and is the method that actually outputs the HTML to the DOM.

Example:

A simple component with a simple `render()` method:

```
class Header extends React.Component {
```

```
  render() {
    return (
      <h1>This is the content of the Header component</h1>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));
```

componentDidMount

The `componentDidMount()` method is called after the component is rendered.

This is where you run statements that requires that the component is already placed in the DOM.

Example:

At first my favorite color is red, but give me a second, and it is yellow instead:

```
class Header extends React.Component {
  constructor(props) {
    super(props);
    this.state = {favoritecolor: "red"};
  }
  componentDidMount() {
    setTimeout(() => {
      this.setState({favoritecolor: "yellow"})
    }, 1000)
  }
  render() {
    return (
      <h1>My Favorite Color is {this.state.favoritecolor}</h1>
    );
  }
}

ReactDOM.render(<Header />, document.getElementById('root'));
```

Unmounting

The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it.

React has only one built-in method that gets called when a component is unmounted:

- `componentWillUnmount()`

---

componentWillUnmount

The `componentWillUnmount` method is called when the component is about to be removed from the DOM.

Example:

Click the button to delete the header:

```
class Container extends React.Component {
  constructor(props) {
    super(props);
    this.state = {show: true};
  }
  delHeader = () => {
    this.setState({show: false});
  }
  render() {
    let myheader;
    if (this.state.show) {
      myheader = <Child />;
    };
    return (
      <div>
      {myheader}
      <button type="button" onClick={this.delHeader}>Delete
Header</button>
      </div>
```

```
    );
  }
}

class Child extends React.Component {
  componentWillUnmount() {
    alert("The component named Header is about to be unmounted.");
  }
  render() {
    return (
      <h1>Hello World!</h1>
    );
  }
}

ReactDOM.render(<Container />, document.getElementById('root'));
```